(12) UK Patent Application (19) GB (11) 2 389 929 (13) A

(43) Date of A Publication 24.12.2003

(21) Application No: 0214414.5

(22) Date of Filing: 21.06.2002

(71) Applicant(s):
Armoursoft Ltd
(Incorporated in the United Kingdom)
1st Floor, Unit 1, Warren Court,
Feldspar Close, Warren Park Way,
Enderby, LEICESTER, LE9 5SD,
United Kingdom

(72) Inventor(s):
Richard Critten
James Edward Charlesworth
Andrew Larter
Peter Green
Tracy John Dale

(74) Agent and/or Address for Service:
Reddie & Grose
16 Theobalds Road, LONDON, WC1X 8PL,
United Kingdom

(51) INT CL[7]:
G06F 12/14 // G06F 1/00

(52) UK CL (Edition V):
G4A AAP A23A

(56) Documents Cited:
EP 1130494 A2 EP 1085420 A1
WO 2002/077747 A2

(58) Field of Search:
UK CL (Edition V) G4A
INT CL[7] G06F
Other: Online: WPI, EPODOC, JAPIO

(54) Abstract Title: Computer Encryption Systems

(57) A system for encrypting computer files associates a key file with the encrypted file. Operations performed on the encrypted file are also performed on the key file. The key file includes a list of users to whom the encrypted file is available together with the public key certificates for those users. The symmetric key is held in the key file wrapped with the authorised user certificates.

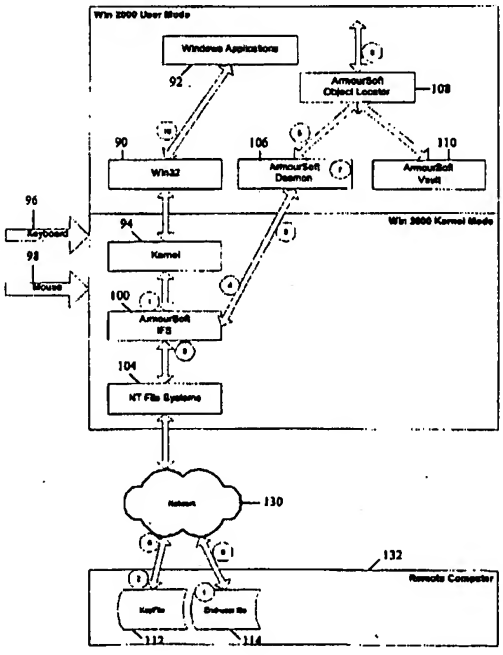

Figure 8

GB 2 389 929 A

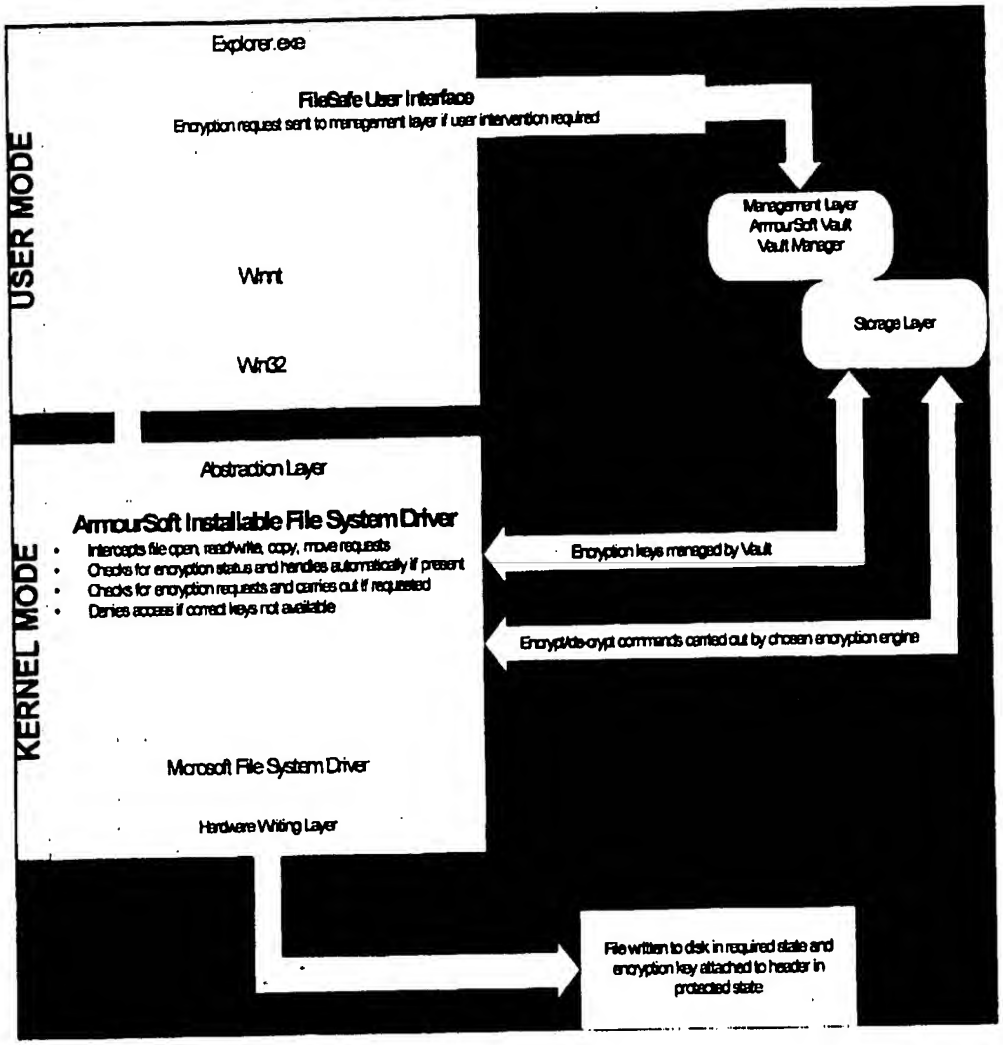
Explorer.exe
FileSafe User Interface
Encryption request sent to management layer if user intervention required
USER MODE
Management Layer
ArmourSoft Vault
Vault Manager
Storage Layer
Win32
Abstraction Layer
ArmourSoft Installable File System Driver
Intercepts file open, read/write, copy, move requests
Checks for encryption status and handles automatically if present
Checks for encryption requests and carries out if requested
Denies access if correct keys not available
Encryption keys managed by Vault
Encrypt/de-cypt commands carried out by chosen encryption engine
KERNEL MODE
Microsoft File System Driver
Hardware Writing Layer
File written to disk in required state and encryption key attached to header in protected state

Figure 1

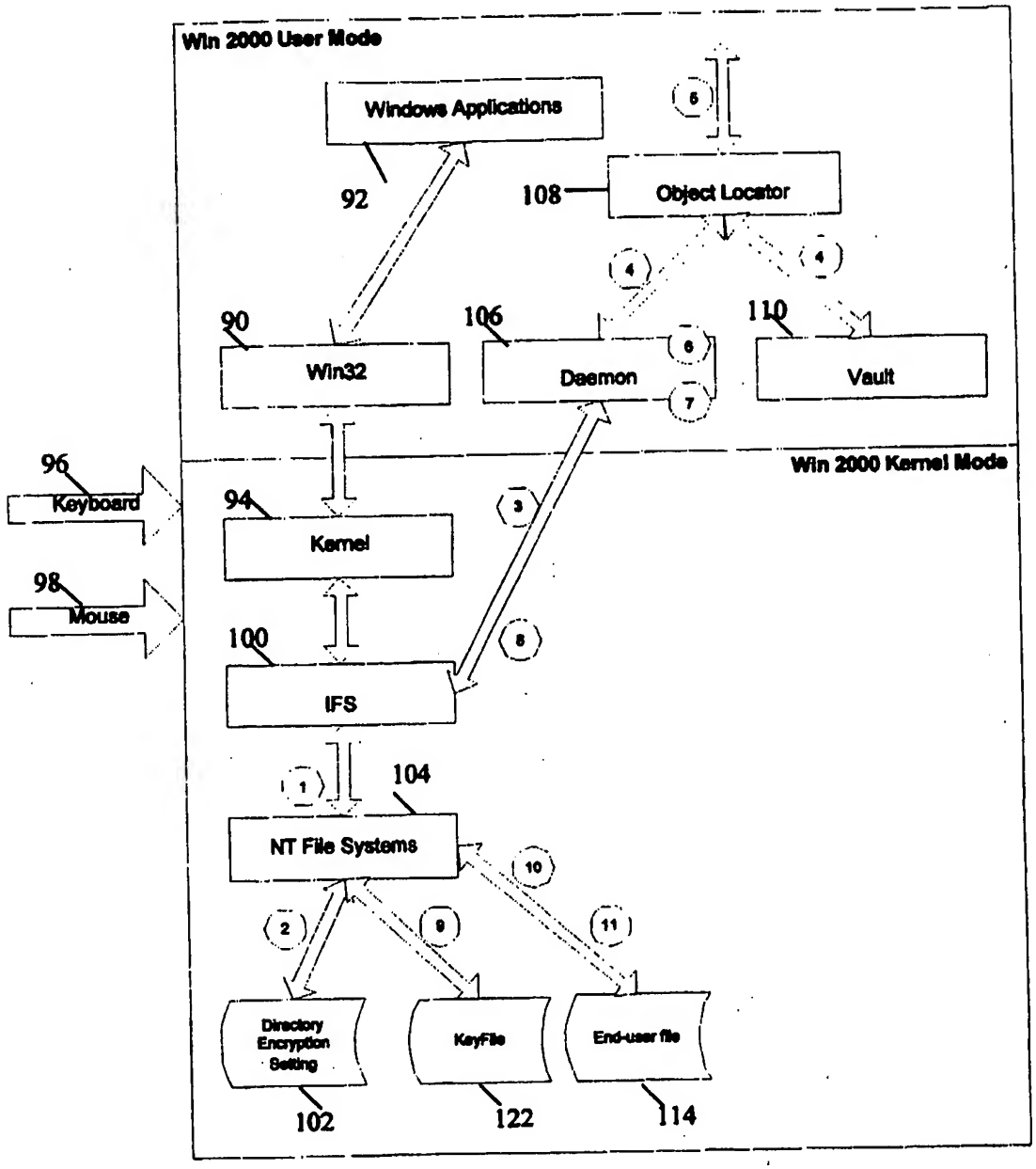
Win 2000 User Mode
Windows Applications
92
108
Object Locator
5
4
4
90
Win32
106
Daemon
6
7
110
Vault
Win 2000 Kernel Mode
96
Keyboard
94
Kernel
3
98
Mouse
100
IFS
8
1
104
NT File Systems
10
2
9
11
Directory Encryption Setting
KeyFile
End-user file
102
122
114

Figure 2

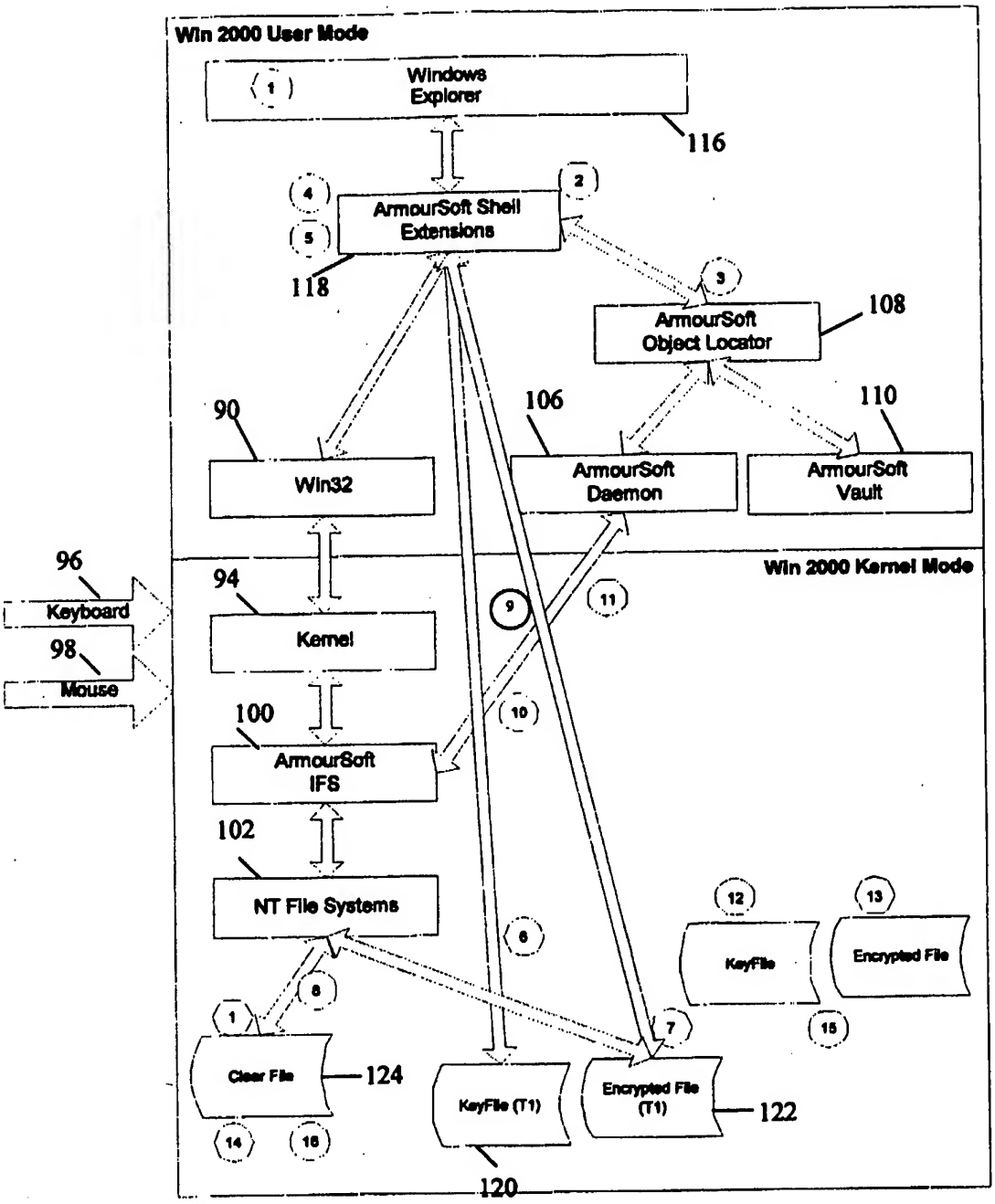
Win 2000 User Mode
Windows Explorer
116
ArmourSoft Shell Extensions
118
ArmourSoft Object Locator
108
Win32
90
ArmourSoft Daemon
106
ArmourSoft Vault
110
Win 2000 Kernel Mode
96
Keyboard
98
Mouse
94
Kernel
100
ArmourSoft IFS
102
NT File Systems
KeyFile
Encrypted File
12
13
15
Clear File
124
KeyFile (T1)
Encrypted File (T1)
122
120
14
16
1
2
3
4
5
6
7
8
9
10
11

Figure 3

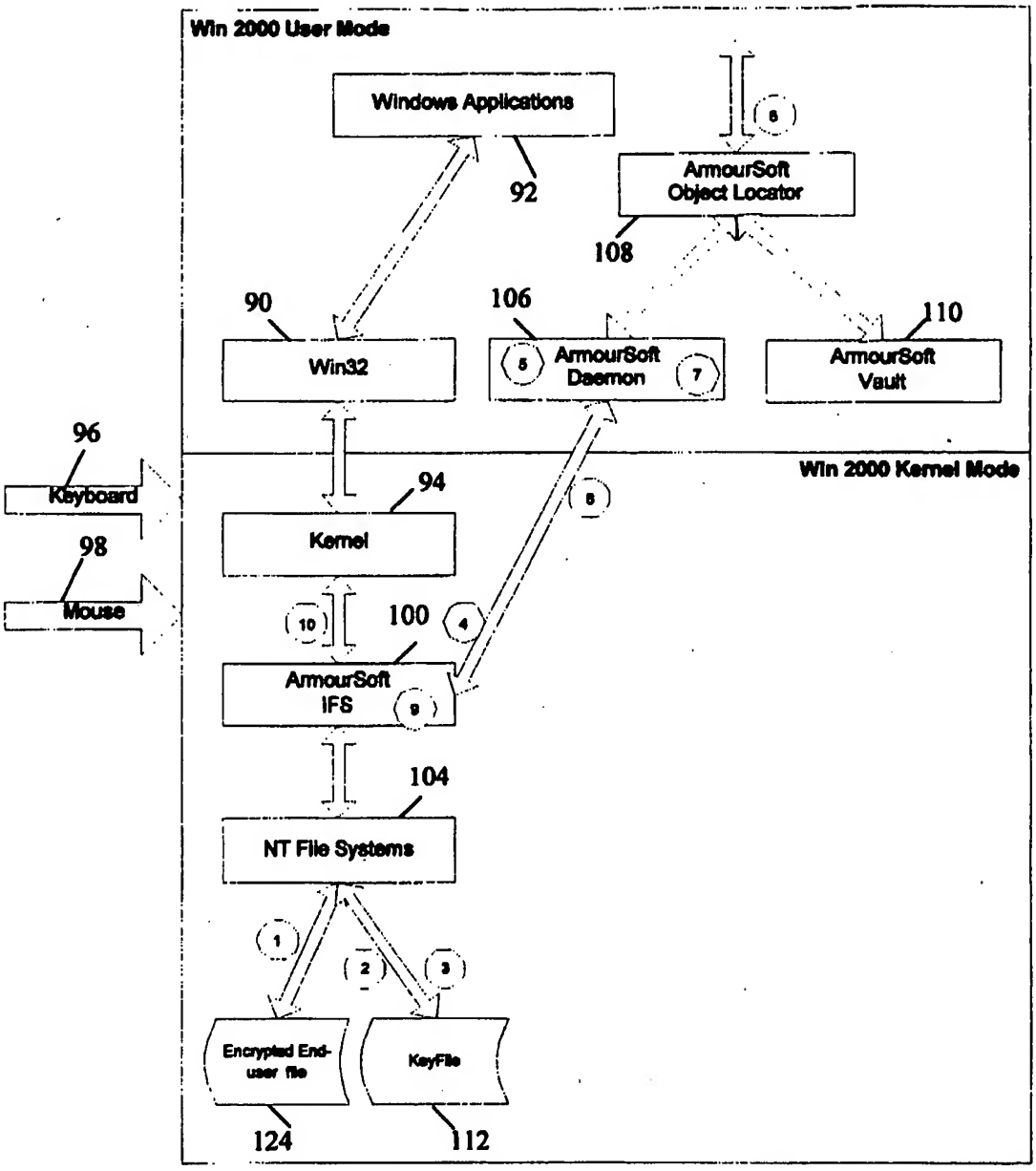
Win 2000 User Mode
Windows Applications
92
ArmourSoft
Object Locator
108
6
90
106
110
Win32
5
ArmourSoft
Daemon
7
ArmourSoft
Vault
96
Keyboard
98
Mouse
Win 2000 Kernel Mode
94
Kernel
8
10
100
4
ArmourSoft
IFS
9
104
NT File Systems
1
2
3
Encrypted End-
user file
KeyFile
124
112

Figure 4

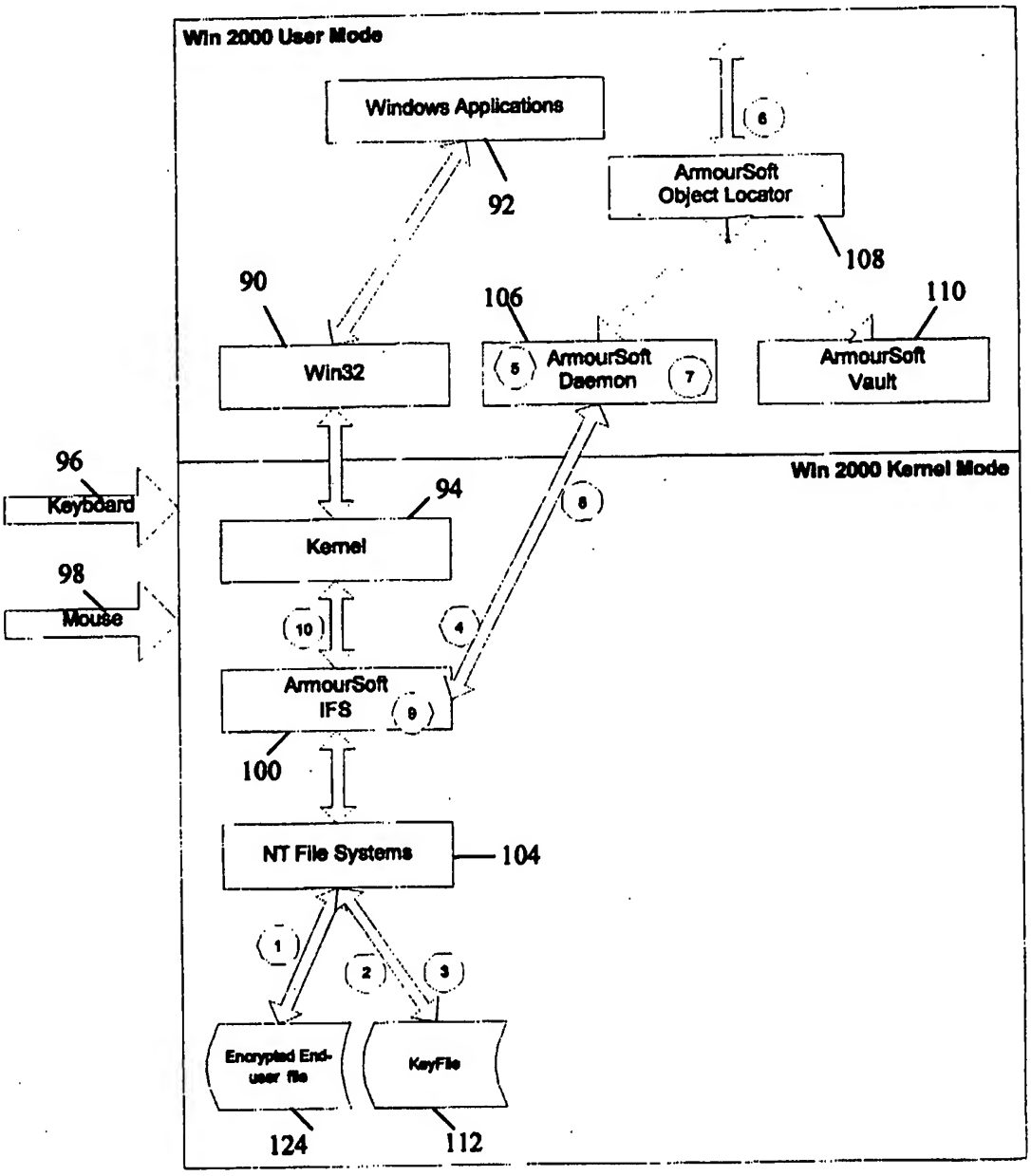
Win 2000 User Mode
Windows Applications
92
6
ArmourSoft
Object Locator
108
90
106
110
Win32
5
ArmourSoft
Daemon
7
ArmourSoft
Vault
Win 2000 Kernel Mode
96
Keyboard
94
8
Kernel
98
Mouse
10
4
ArmourSoft
IFS
9
100
NT File Systems
104
1
2
3
Encrypted End-
user file
KeyFile
124
112

Figure 5

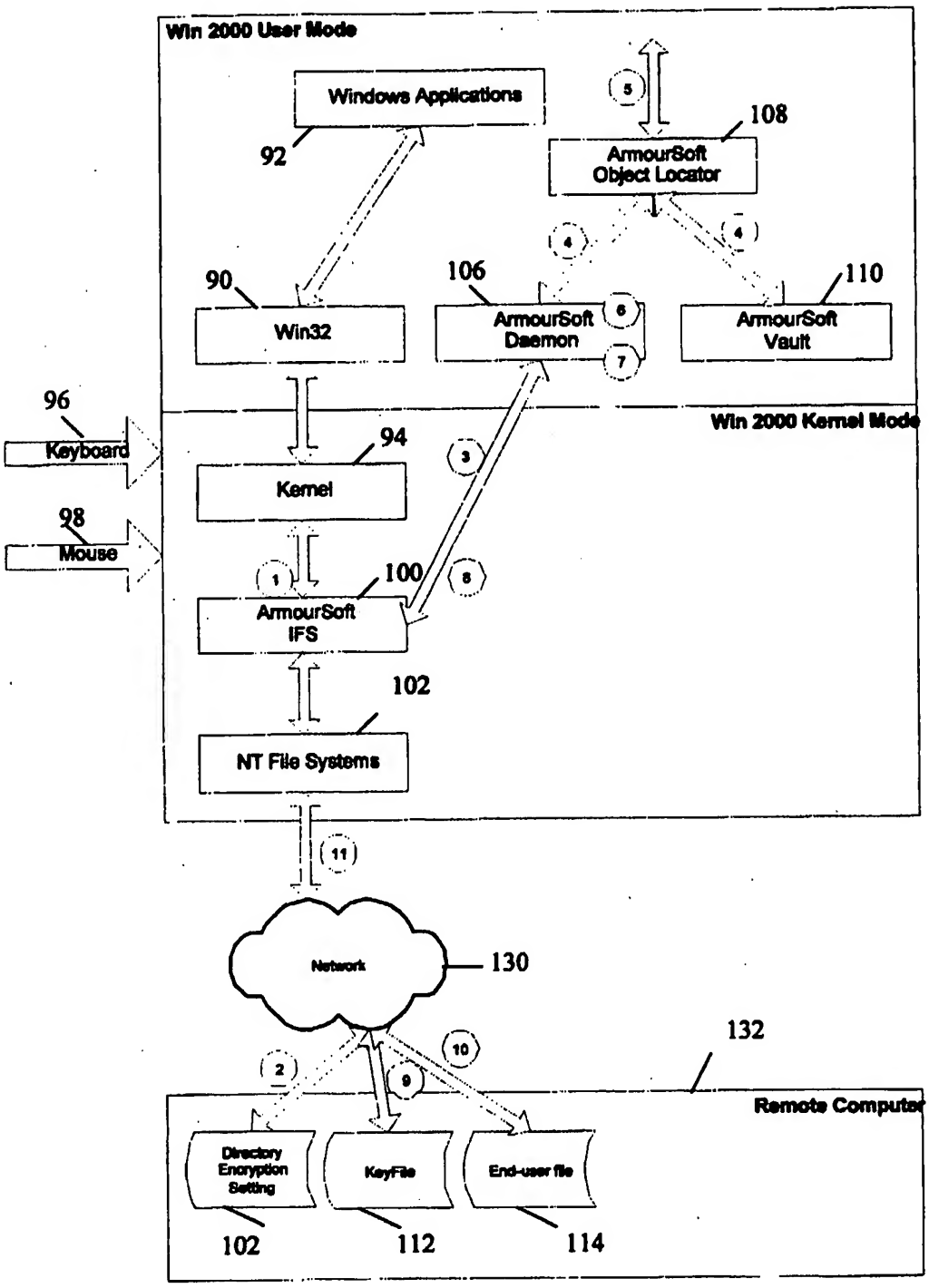
Win 2000 User Mode
Windows Applications
92
5
108
ArmourSoft Object Locator
4
4
90
Win32
106
ArmourSoft Daemon
6
7
110
ArmourSoft Vault
96
Keyboard
98
Mouse
Win 2000 Kernel Mode
94
Kernel
3
1
100
8
ArmourSoft IFS
102
NT File Systems
11
Network
130
132
10
2
9
Remote Computer
Directory Encryption Setting
KeyFile
End-user file
102
112
114

Figure 6

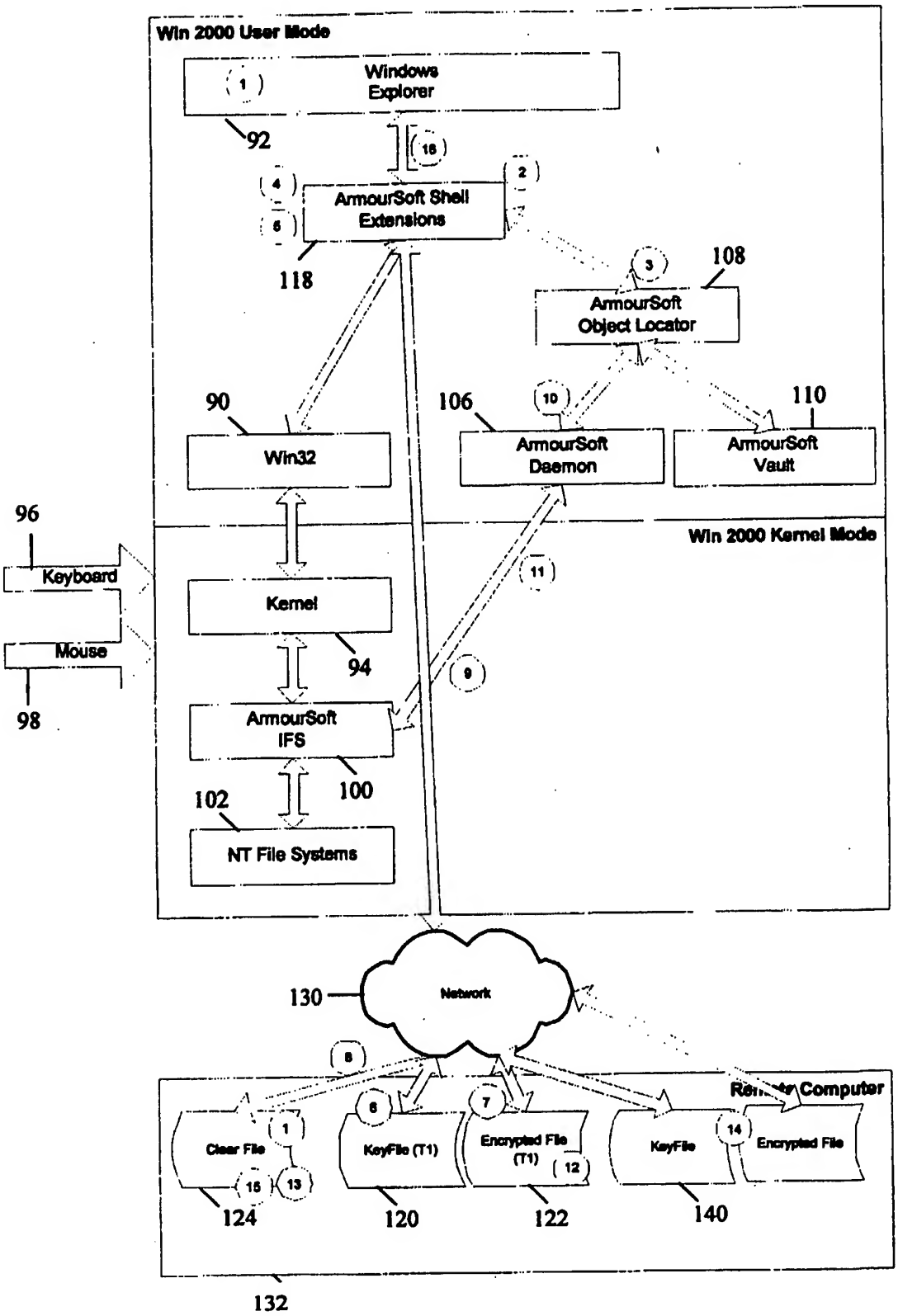
Win 2000 User Mode
Windows Explorer
92
ArmourSoft Shell Extensions
118
ArmourSoft Object Locator
108
90
Win32
106
ArmourSoft Daemon
110
ArmourSoft Vault
Win 2000 Kernel Mode
96
Keyboard
Mouse
98
Kernel
94
ArmourSoft IFS
100
102
NT File Systems
130
Network
Clear File
124
KeyFile (T1)
120
Encrypted File (T1)
122
KeyFile
140
Encrypted File
132

Figure 7

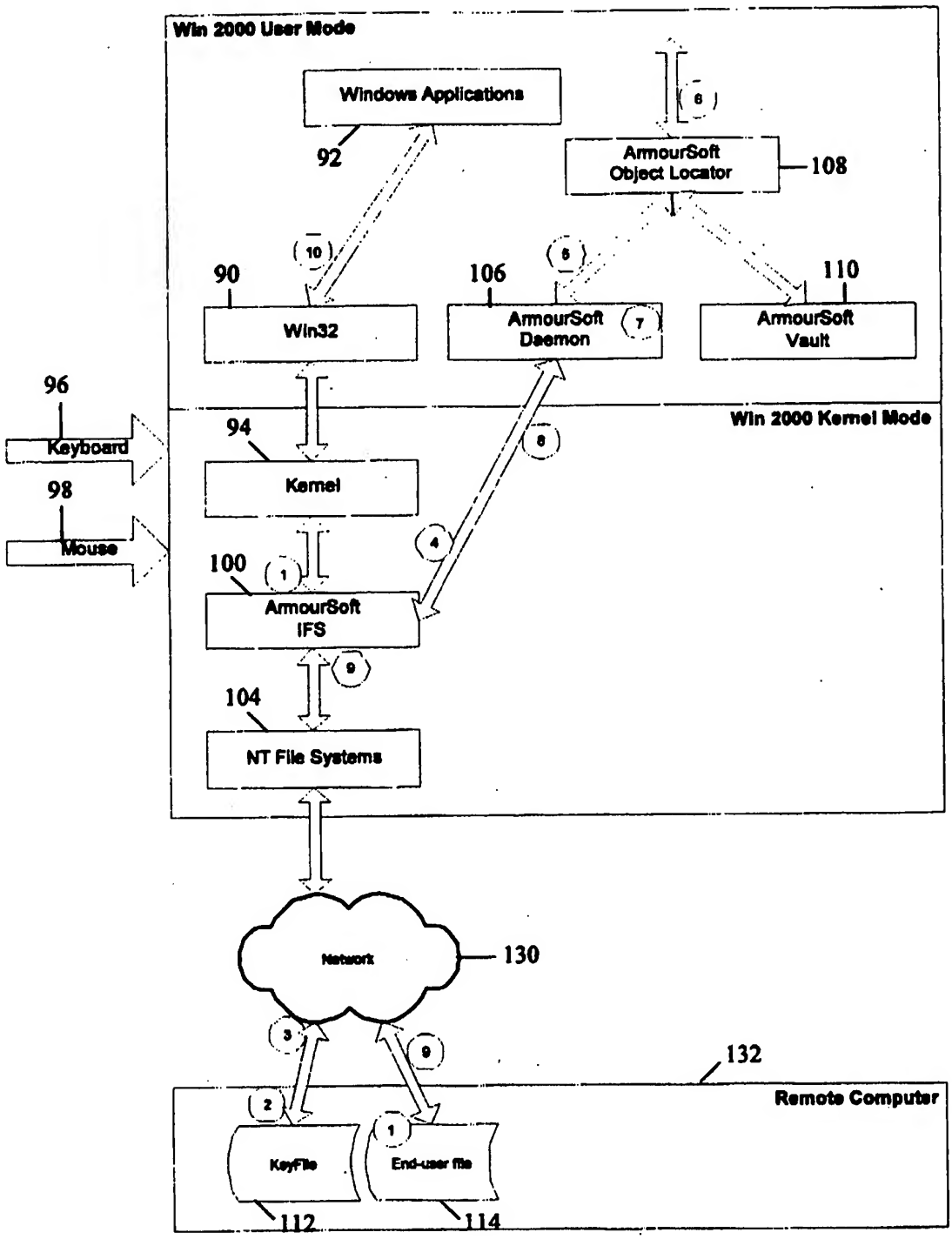
Win 2000 User Mode
Windows Applications
92
ArmourSoft
Object Locator
108
10
6
90
106
110
Win32
ArmourSoft
Daemon
7
ArmourSoft
Vault
96
Keyboard
98
Mouse
Win 2000 Kernel Mode
94
Kernel
8
4
100
1
ArmourSoft
IFS
9
104
NT File Systems
Network
130
3
9
132
Remote Computer
2
1
KeyFile
End-user file
112
114

Figure 8

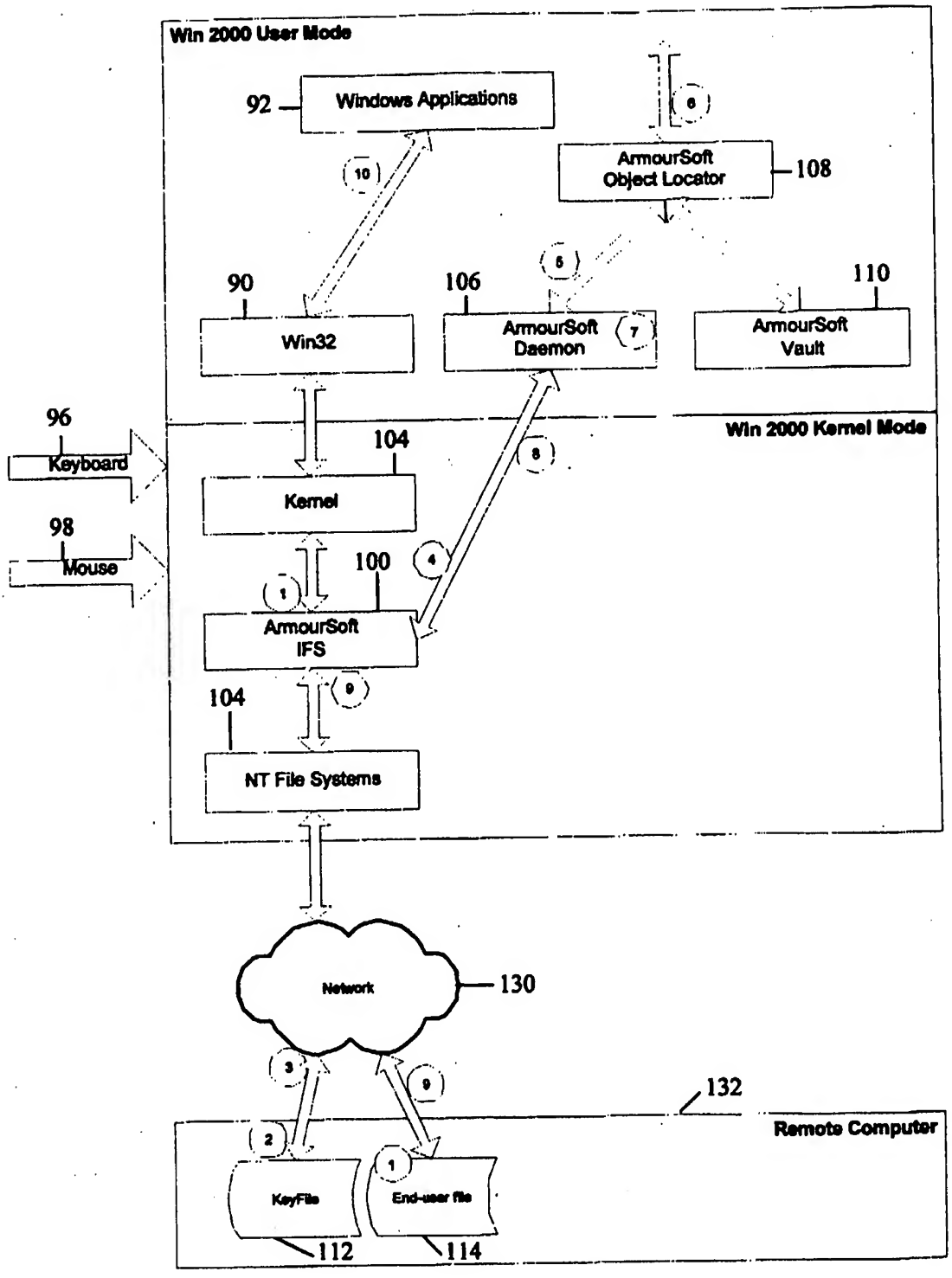
Win 2000 User Mode
92
Windows Applications
6
ArmourSoft
Object Locator
108
10
90
106
110
5
Win32
ArmourSoft
Daemon
7
ArmourSoft
Vault
96
Keyboard
104
Win 2000 Kernel Mode
8
Kernel
98
Mouse
100
4
1
ArmourSoft
IFS
104
9
NT File Systems
Network
130
3
9
132
Remote Computer
2
1
KeyFile
End-user file
112
114

Figure 9

Computer Encryption Systems

This invention relates to computer encryption systems.

Computer encryption systems known in the art fall broadly into two categories: hard disk or partition-based systems, and file-based systems. Each type has associated advantages and disadvantages.

Hard disk-based systems have the advantage that the entire disk is guaranteed to be encrypted. Within the disk there can be no leaked clear files which may not have been encrypted. Furthermore, if the system partition is encrypted then the hard disk encryption system must obtain the key from the end user before the computer is able to boot.

Hard disk-based systems have the disadvantage that the encryption driver must be running on the computer before the operating system has loaded. Thus, the encryption driver cannot access operating system resources such as the GUI interface and hardware that needs operating system drivers. For the same reason external secure media such as smart cards cannot be accessed without the hard disk encryption system developer writing specific code to support the smart media and any other system device the smart media needs to be able to function.

A weakness in hard disk-based systems is that the protection key is only protected by the users' pass phrase. These cannot be stored on smart media and so are vulnerable to social engineering attacks. Hard disk encryption systems are also time consuming to install. As the whole disk has to be encrypted, the encryption process may take hours during which the computer cannot be used.

In file-based systems, only those files and folders that need encryption are encrypted. This reduces the time required for encryption and takes place after the operating system has loaded and the end user has been authenticated. As a result, any smart media, readers, and other hardware that are supported by the operating system can be used by

the encryption process. Moreover, there is no post installation encryption phase as required by the hard disk encryption method. New files may be encrypted automatically and existing clear files that require encryption can be encrypted either individually or in bulk.

File based systems have the disadvantage that there is always a potential leakage of unencrypted files. These may be temporary files the permanent versions of which will be encrypted. This problem can usually be avoided by a correct set up of the computer and encryption system.

It will be appreciated from the above discussion that both systems have advantages and disadvantages. However, neither permit end users to share access to their encrypted files without releasing the key used to encrypt the file. Neither can they allow users to grant encrypted access to files to other users without access to the other user's key.

In all areas of work, people work in groups and collaborate. They need to share files and data. To maintain security whilst allowing this type of access, a system must permit end-users to encrypt their files, allow their encrypted files to be accessed by others and work together on a group of encrypted files. Organisations maintaining and operating the computer systems need to be able to recover encrypted files when the end-user is no longer available, for example they have left the company.

The invention aims to address the problem with the prior art discussed above and to provide a system and method which can meet the requirements discussed.

Broadly, the invention uses a key file associated with a file to be encrypted. The key file holds the decryption key and all file handling, such as moving, copying etc. performed on the file to be encrypted is also performed on the key file. The key file may hold decryption keys for a number of users making multiple access possible.

More specifically there is provided a method for encrypting computer files comprising: encrypting at least one file within a folder; for each encrypted file, creating

an associated key file containing the decryption key, wherein the association between an encrypted file and its key file is such that any file handling operation performed on the encrypted file is also performed on the key file.

In one preferred embodiment, folders are marked to indicate that files within that folder are to be encrypted. Preferably a file is placed in the folder containing default encryption setting for files created in the folder.

Preferably, the associated key files are hidden from a user by a file system driver. The file system driver also ensures that when an operation is performed on a file, the same operation is performed on the key file. The file and the key file may have the same file name but different extensions.

In one preferred embodiment of the invention the key file contains decryption keys for a plurality of users. This allows an encrypted file to be made available to a plurality of users.

Preferably, the key file comprises a wrapped symmetric key for the associated encrypted file. The symmetric key may be wrapped by an end-user's public key.

Preferably, the default encryption settings include a list of end-users for which a file is to be encrypted.

Preferably, on receipt of a file encryption request, the file system driver reads the directory encryption settings file and retrieves the end-user list for which the file needs to be encrypted. The certificates for all the end-users on the list are located and the key file is created from the symmetric file encryption key wrapped end-user's by the public key for each end-user in the end-user list.

Where a file is to be encrypted in a directory that has not been marked for encryption, on receipt of a request to encrypt a file, the end-user may be asked for a list of end-users for whom the file is to be encrypted, the certificates for each end-user on the list are then retrieved, and a symmetric file encryption key generated. The file

encryption key is then wrapped with the certificate for each user, the wrapped encryption keys comprising the key file.

In this preferred embodiment, the step of encrypting the file may include saving the file in a temporary file to the name of the key file. The symmetric file key may then be requested and the original clear file read. The temporary encrypted temporary file may then be written and the clear file renamed to a further temporary file. The encrypted temporary file is then renamed to the original file name and the further temporary file deleted.

Embodiments of the invention may be used to encrypt files over a network. In that case, in one embodiment the directory encryption settings file is stored in a network folder and an encrypted network file is created at a local user computer and sent over the network in encrypted form.

In another case, the encrypted temporary file is stored in a network folder, and the further temporary file is stored at a local computer, whereby the clear file is not transmitted over the network.

The invention also provides a method of reading or writing a file encrypted according to the method defined above, comprising detecting the key file associated with the encrypted file, retrieving the decryption key from the key file, decrypting the file using the decryption key, and reading and/or writing the decrypted file.

Preferably, retrieving the decryption key comprises finding a private key, and using the private key to unwrap a wrapped key in the key file. Where the file has been encrypted for group access, the key file may include a list of users for whom the file is encrypted. Finding a private key then comprises finding a private key that corresponds to the list of users.

In one preferred embodiment, the key file is a network key file, the key file is opened at the local user's computer and the file is decrypted at the local computer after is has been read across the network. This has the advantage that

the file is always transported across the network in encrypted form so maintaining security.

The invention also provides a system for encrypting computer files comprising an installable file system driver transparent to the computer operating system for reading directory encryption settings and retrieving an end user list for which the file is to be encrypted, a daemon, a vault, the daemon operating to retrieve certificates stored in the vault, means for generating a symmetric encryption key, the key being sent by the daemon to the file system driver, the file system driver including means for creating a key file from the end-user certificates and the symmetrical encryption key and for associating the key file with the encrypted file.

Preferably, the system includes an object locator, wherein the object locator communicates between the vault and the daemon for retrieval of certificates. The vault may comprise a smart media and/or a soft vault on an end-user's computer.

A preferred embodiment of the invention comprises an operating system shell extension for retrieving certificates, generating the symmetric file encryption key and wrapping the symmetric key with the certificate of each user who has access to an encrypted file.

Embodiments of the invention use file or folder-based encryption. This means that the encryption within a computer system is selected. Such a system can use any smart media that is supported by the operating system. By allowing the computer to boot before accessing the smart-media, no manufacture specific interface need be written to access the smart media. This allows industry standard APIs to be used. Thus, the end user has the choice of any smart media supported by the operating system and can choose the smart media to be compatible with other applications required by the system.

This sort of flexibility would not be possible in a hard disk-based encryption system. In these systems, when

the computer boots, the encryption needs the decryption key before the operating system can be loaded. The device drivers needed to provide access to smart media or files on a card disk have not yet been loaded and so are not available to the encryption system. As a result, these systems must rely on pass phrase-based protection or can only support a limited number or range of smart media products.

The key files generated are not visible to the user and are associated with the encrypted files such that the two are always operated on together. The key files are small and so do not significantly affect the total file size or response times.

Whilst the advantages of the invention have been discussed in conjunction with smart media, end-users do not always want to use smart media to gain access to systems. For example, they may consider that the cost of smart media security is not justified in their environment. Embodiments of the present invention use a soft vault which is stored on the end-user computer. Preferably, this vault will have many of the characteristics of smart media. The file-based encryption system can use this vault instead of or in addition to smart media.

Embodiments of the invention will now be described, by way of example, and with reference to the accompanying drawings, in which:

Figure 1 shows how an embodiment of the invention may be integrated into a Microsoft Windows (RTM) operating system such as Windows 2000;

Figure 2 is a schematic diagram showing how an encrypted file is created in a folder marked for encryption;

Figure 3 is a schematic diagram showing how an encrypted file is created from an existing unencrypted File:

Figure 4 is a schematic diagram showing how an existing encrypted file can be read;

Figure 5 is a schematic diagram showing how an existing encrypted file can be written;

Figure 6 is a schematic diagram showing how a new encrypted file can be created across a network in a folder marked for encryption;

Figure 7 is a schematic diagram showing how a new encrypted file can be created across a network from an existing unencrypted File:

Figure 8 is a schematic diagram showing how an existing encrypted file can be read across a network; and

Figure 9 is a schematic diagram showing how an existing encrypted file can be written across a network.

The embodiments to be described use a file-based encryption system. This allows smart-media to be used in the system simply. End user files can be protected by a combination of the smart media and the PIN (Personal Identification Number) for the smart media. In practice, the PIN is not necessarily a number but, for example, a series of alphanumeric or other characters. A soft vault internal to the system may be used instead of the smart media in order to reduce costs.

The system to be described allows file sharing by different users. To achieve this using a single user encryption system, that is where the file is originally encrypted by a single user, the end-user files are first decrypted for the first user and then re-encrypted for the second user. End-user files must be clear, or unencrypted, to be shared and another copy made for the second, or each further, user. Such an approach makes collaboration difficult. The embodiment to be described allows simultaneous encryption of end-user files for multiple users without an increase in file size allowing authorised users to access the same files without copies having to be made for each user. Thus, efficient and secure multi-user and group encryption is achieved.

Embodiments to be described also permit encryption across a network. The encryption system sits above the operating file system drivers on the computer system and so has transparent access both to files on the local computer

and the network allowing files available on the network to be encrypted, shared by group encryption and have recovery encryption applied to them. The encryption and decryption take place on the computer local to the user. This means that files cross the network in encrypted form so there is no vulnerability to a clear file being intercepted on the network.

Embodiments of the present invention use an Installable File System (IFS) that acts as a low-level file driver. When handling encrypting files, it acts as a virtual file system handling all file operation transparently to the operating system, applications and users.

Figure 1 illustrates how the system may integrate into an operating system such as Windows 2000.

The manner in which files are encrypted will now be described.

Applications are provided which set a default file and folder encryption settings for the computer and for each folder. The computer default application stores default settings for encrypting new files and folders centrally on the computer. The stored information includes the identity of the user or users for which new folders or files should be encrypted. This information can be pre-loaded on the computer when it is first set up or by the end-user at some later time. This application accesses end-user certificates and presents a list of available users for selection. Once selected, the certificate identities are stored on the computer as the default encryption settings. These settings may be imported or exported to other computers allowing quick configuration.

The default encryption settings for the computer are stored in the computer's registry. The format of these settings is:

```
DefaultEncryptionSettings\{identity}
    EntityIdentifer={identity}
    Description={end-user description}
```

RecoveringKey={identity}

The above settings then repeated for each identity used for default encryption.

The application that sets the folder settings stores the folder settings on the end-user's computer. New files are then automatically encrypted. On set up of the default conditions for a folder, the user is presented with the default encryption settings for the computer. The end-user may modify this list to add or remove users by selecting their certificates. The settings may be propagated down a tree of folders.

Folders on an end-users computer system are not in themselves encrypted. Instead they are marked to indicate then any files created in them should be created encrypted. This marking is done by placing a file

ASEF_Dir_Default.dat

into the folder. The file is hidden from the end-user by the system driver. The file contains the default encryption setting for all new files created in the folder. The contents of the file are formatted as follows:

ASEF_Dir_Default.dat:	<SuniveralData_Header> <SDirectorySettingsFile_V1_HeaderInfo> <SUniversalData_V1_EncryptionInfo> <encrypting_entity_collection> <padding> <SUniversalData_V1_Checksum>
SUniversalData_Header:	<Identifier> <Version>
SDirectorySettingsFile_V1_HeaderInfo:	<TotalFileLength>

SUniversalData_V1_EncrpytionInfo:	<EncryptionAlgorithm>
Encrpyting_entity_collection:	[<encrypting_entity>] <encrypting_entity_sentinal>
Encrypting_entity:	<SUniversalData_V1_EntityInfo> <SEncrpytedFile_V1_Entity_User>
SUniversal_V1_EntityInfo:	<Entity Type>
SEncrpytedFile_V1_Entity_User:	<UserIDLength> <UserNameLength> <KeyBlockLength> <UserID> <UserName> <KeyBlock>
Identifier:	<integer>
Version:	<integer>
TotalFileLength:	<long integer>
EntityType:	<integer>
UserIDLength:	<integer>
UserIDLength:	<integer>
KeyBlockLength:	<integer>
UserID:	[<byte>]
UserName	[<byte>]
Keyblock:	[<byte>]

A group-based encryption setting may be created by creating an identity for a group of users and providing this group with a private key for the group, as well as their own

key. This effects a simple, but powerful group encryption scheme. By ensuring that all files to which members of the group require access are included, in folders that include the group identity in the default settings, files in these folders will be encrypted for the group. Files are encrypted for all users in the folder's or computer's encryption settings.

When files are encrypted the decryption key for each user is stored in a companion file with a known but different file extension, for example if MyData.doc is encrypted then the key file MyData.as_enc is created to hold the wrapped encryption/decryption keys. The file system driver hides this file from the end-user.

If the end-user's file is moved, copied or renamed then the file system driver moves, copies or renames the file as appropriate. Thus, any file handling operation performed on the end-user's file is also performed on the associated key file.

The contents of the file are formatted as follows:

Key_file.as_enc	<SUniversalData_Header><SEncryptedFile_V1_HeaderInfo><SUniversalData_V1_EncryptionInfo><encrypting_entity_collection><padding><SUniversalData_V1_Checksum>
SUniversalData_Header	<Identifier><Version>
SEncryptedFile_V1_HeaderInfo	<TotalFileLength>
SUniversalData_V1_EncryptionInfo	<EncryptionAlgorithm>

encrypting_entity_collection	[<encrypting_entity>]<encrypting_entity_sentinal>
Encrypting_entity:	<SUniversalData_V1_EntityInfo><SEncryptedFile_V1_Entity_User>
SUniversalData_V1_EntityInfo:	<EntityType>
SEncryptedFile_V1_Entity_User:	<UserIDLength><UserNameLength><KeyBlockLength><UserID><UserName><KeyBlock>
Identifier:	<integer>
Version:	<integer>
TotalFileLength:	<long integer>
EntityType:	<integer>
UserIDLength:	<integer>
UserIDLength:	<integer>
KeyBlockLength:	<integer>
UserID:	[<byte>]
UserName	[<byte>]
KeyBlock:	[<byte>]

The file system driver creates this file whenever a file is created in a folder marked for encryption.

The KeyBlock is the wrapped symmetric key for the file. The symmetric key is wrapped by an end-users public key (contained in a certificate) and can only be decrypted by the user's private key. As public certificates are available to all end-users, any end-user may encrypt a file for any other end-user providing the first end-user can decrypt the file in the first place. Once the first end-user using their private key has managed to decrypt their wrapped key

they may wrap the symmetric file key for any other end-user and append a new encrypting_entity to the encryption file.

The group-based encryption may be used to implement a file recovery scheme. A recovery user identity (key pair and certificate) is created and this identity identifier is included on all a computer's default encryption settings. Any encrypted files created will automatically be encrypted for this recovery user. If access is required to files that have been encrypted, but for which the end-user's key is not available, the recovery key can be used in its place.

When a file is encrypted, a key file is created to hold the wrapped encryption/decryption keys for the file. This key file has the same name but a different extension from the end-user file. When the end-user file is moved, copies, deleted or renamed, the key file is moved, deleted or renamed as appropriate. The encryption system file system driver treats the two files as if they were one. The user is not aware of the second file containing the wrapped keys.

When a new file is created in an encrypted folder, the end-user needs to have certificates available for all end-user identities in the default setting for the folder. If these are all available, the new file is created using the keys from the end user certificates. If any certificates are missing, the user is prompted to retrieve them.

Having selected an existing file, the end-user may request that the file is encrypted. The default encryption settings for the folder are used or if the folder is not encrypted then the default settings for the computer are used. In both cases the end-user needs to have available the certificates for all the end-user identities contained in the default settings (folder or computer as appropriate). If all these certificates are available (which they normally would be) then the file is encrypted for the list of end-users. If any certificates are not presently available then the end-user is prompted to try to retrieve the missing certificates. This may be done in a variety of methods. If

the end-user still cannot provide all the required certificates the file may optionally still be encrypted.

Having selected an existing encrypted file the end-user may request for the file to be decrypted. To achieve this, the end-user must have available the private key corresponding to any of the identities used to encrypt the file. If one of these private keys is available the file is decrypted. If none of the corresponding private keys is available to the end-user, the end-user is prompted to retrieve one of the required keys. If one of the required private keys can be made available, the file is decrypted. If however none of the private keys can be obtained by the end-user then access is denied to the file and the file is not decrypted.

Third party applications (e.g. MS-Word) create, delete, read and write files. When performing these file operations on encrypted files or creating new files in encrypted folders the user is not aware that the encryption/decryption process is happening. The end-user is only required to intervene if the encryption system does not have all the required keys available. When the third party application tries the open an encrypted file, the file is decrypted as it is read from the disk and into memory. To achieve this, the end-user must have the private key corresponding to any of the identities used to encrypt the file. If one of these private keys is available then the file is decrypted as it is read into memory by the application. If none of the corresponding private keys is available to the end-user, the end-user is prompted to retrieve one of the required keys. If one of the required private keys can be made available, the file is decrypted as it is read into memory by the application, if however none of the private keys can be obtained by the end-user then access is denied to the file and the file is not decrypted. The third party application will usually present the end-user with an access denied message; obviously the details of the message depend upon

the application.

Once the file has been opened the application can continue to read from or write to the file without further intervention being required from the end-user.

By way of a further example, figures 2 to 9 describe some of the above-mentioned processes in greater detail.

Referring now to figure 2, the manner in which a new encrypted file is created in a folder marked for encryption will be described. Figure 2 shows Windows 2000 user mode with the WIN32 operating 90 system communicating with Windows applications 92 and with the kernel 94 in kernel mode. The system in kernel mode can accept external user inputs via a keyboard 96 and a mouse 98. These are, of course, only exemplary and other data inputs may be provided.

At 1, the file system driver embodying the invention 100 receives a >create= request for the end-user file. The request asks for an encrypted file to be created. At 2, the file system driver 100 reads the directory encryption settings 102 via the Windows NT File Systems 104 and retrieves the list of end users for which the file needs to be encrypted. At 3, the IFS driver 100 transmits this list of users to a daemon 106 which, at step 4 uses the system object locator 108 to try and find all the certificates for the end users on the list from the system vault 110. The daemon is a process running in the background that performs operations in response to certain events. If any certificates are missing, at step 5, the object locator 108 prompts the user to find any certificates that the object locator has not been able to retrieve.

Once all the certificates are present, the daemon 106 generates the symmetric file encryption key at step 6. At step 7, the daemon wraps the file encryption key for each user in the user list using the corresponding certificate which provides the public key. At step 8, the daemon 106

returns the file encryption key with the list of users with the wrapped keys attached to the file system driver 100. The driver 100 then at step 9 uses the user list and the wrapped keys to create a key file 112 for the end-user file being created. Following this the end user file 114 is created at step 10 and at step 11 any data written to the end user file 114 is encrypted with the file encryption key.

Figure 3 is a similar diagram to figure 2 showing how a new encrypted file can be created from an existing unencrypted file.

Initially, the user selects, at step 1, the file to be encrypted. This may be done, for example, via Windows Explorer 116. At step 2, a system shell extension 118 prompts the end user to choose the end-users for which the file is to be encrypted. It is important to understand that the file may be encrypted for a single user or a group of users.

At step 3, the shell extension retrieves certificates for the users chosen by the end user. If the certificates cannot be retrieved, the shell extension prompts the user to provide them. At step 4, the shell extension then generates the symmetric file encryption key, which it wraps at step 5, for each selected end-user. The shell extension then writes at step 6, the key file to the system disk with a temporary name (T_1) 120. At step 7 the file originally selected in Windows Explorer 116 is copied to the temporary file name T_1 which causes the file system driver 100 to encrypt the file to create an encrypted file (T_2) 122. The original clear file 124 is then by the file system driver 100 at step 8. The driver 100 sees that the destination file is encrypted and requests, at step 9, the symmetric file key from the daemon 106. The daemon retrieves the file key from the shell extension 118 and, at step 11, returns the key to the file system driver 100. The driver 100 can then read the original clear file and writes the new encrypted temporary file T_1. At step 13, the original clear file is renamed to a new

temporary file T_2 by the shell extension 118 and at step 14 the encrypted file is renamed by the shell extension. The user now has the original file encrypted saved as T_1 which is then saved back to its original name at step 15. When this is successfully completed, the second temporary file, T_2, is deleted leaving the user with just the original file encrypted.

Figure 4 shows how an existing encrypted file may be read. The file system driver 100 receives a request from the user to open a file, for example by a combination of the mouse 98, and a Windows application 92. The file system driver 100 detects that a key file exists for the file that has been requested and therefore that the file is encrypted (step 2). The file system drive opens the key file and retrieves and reads the list of users contained within it at step 3 and, at step 4, passes the retrieved user list together with the wrapped keys contained in the key file to the daemon 106. The daemon at step 5 searches for a private key that corresponds to the user list received from the file system driver. This search is conducted via the object locator 108 and the vault 110. At step 6, if the private key is not found by the daemon, a prompt is sent to the user to provide a private key. When the private key has been found, at step 7, the daemon unwraps the wrapped key corresponding to the retrieved private key. The step 8 this unwrapped key is returned to the file system driver 100 which, at step 9, uses the unwrapped key to decrypt the encrypted file 124 and return a clear file to the application via the kernel 94 at step 10.

The routine shown in figure 5 writes to an existing encrypted file. The routine is identical to steps 1 to 9 of the encrypted file routine described above with respect to figure 4 above. Once the unwrapped key has been used to decrypt the file at step 9, the application can write clear data to the file. This is then encrypted by the file system drive 100 and encrypted data can be written to the file.

The routines shown in figures 6 to 9 correspond to figures 2 to 5 where files are encrypted or decrypted across a network. In the example of figure 6 a new encrypted file is created at a remote computer 132 across a network 130 in a folder marked for encryption. At step 1, the file system driver 100 at a local computer receives a create request for the end-user file to be created over the network on another computer, shown in the figure as the remote computer 132. The end-user file is shown as file 114 at the remote computer 132. The file system driver 100 on the local computer at step 2 reads the directory encryption settings file 102 from the network folder and retrieves the list of end-users for which the network file needs to be encrypted. At step 3, the file system driver 100 on the local computer sends this user list to the daemon 106 on the local computer. The daemon, at step 4, uses the object locator 108 and the vault 110 to retrieve the certificates for the users on the user list. If those certificates are not all received, the user, at step 5, is prompted for the missing certificates. The daemon at step 6 then generates the symmetric file encryption key and at step 7 wraps the file encryption key for each user in the user list using the corresponding certificate, that is the public key. At step 8, the file encryption key and the user list, together with the wrapped keys is sent back from the daemon 106 on the local computer to the local computer file system driver 100. The driver 100 then creates the key file 112 from the user list and the wrapped keys in the network folder for the end-user remote file being created. The local system driver then creates the end-user network file 114 over the network. Data now written to the network file is encrypted with the file encryption key by the local file system driver. All data sent over the network is encrypted.

Figure 7 is a network version of figure 3 where a new encrypted file is created at the remote computer from an existing, clear, file residing on that remote computer. The

end-user at the local computer selects a file to be encrypted, for example via Windows Explorer 116 at step 1. At step 2, the shell extension 118 at the local computer prompts the user to choose the end-users for whom the file is to be encrypted. At step 3, the shell extension 118 retrieves certificates for the selected users, using the object locator 108 and the vault 110. If they cannot be found, the local computer user is prompted to provide the certificates. At step 4, the shell extension generates the symmetric file extension key and at step 5, the local computer shell extension wraps the file encryption key for each selected end user.

At step 6, the shell extensions 118 on the local computer writes the key file 120 to the network folder with a temporary name T_1. At step 7, the shell extension 118 copies the end-user network file originally selected by the user to the temporary file name T_1 in the network folder. The new network file T_1 is then encrypted by the local computer file system driver. The clear original network file is then read at step 8 by the file system driver which, at step 9, sees that the destination network file is encrypted and requests the symmetric file key from the local daemon 106. At step 10, the local computer daemon retrieves the file key from the local shell extension 118 and at step 9 passes the key to the local computer file system driver 100. At step 12, the file system driver 100 on the local computer reads the original clear network file and writes the new encrypted temporary network file T_1. At step 13, when the file is completely written, the original clear network file is renamed to a new temporary network file T_2. The shell extension 118 on the local computer renames the encrypted network file T_1, back to the original network file name at step 14. Once this has been completed, at step 15, the second temporary network file (the clear file) is deleted and at step 16 the end user is left with the original network file encrypted.

Figure 8 shows how an existing encrypted file may be read across a network. At step 1 the local computer file system driver receives a request to open a network file. The file system driver 100 at step 2 detects that there is a network key file for the file to be opened, and that, therefore, the file is encrypted. The file system driver opens the network key file 112 at step 3 and reads the list of users over the network. This list is passed by the file system driver 100 at step 4 to the local system daemon 106 together with the wrapped keys. At step 5 the local daemon tries to find a private key that corresponds to the list of users passed to it by the local file system driver. At step 6, the end user is prompted, on the local computer for a private key if no private key can be found. At step 7, when a private key has been found, the local computer daemon unwraps the wrapped symmetric key that corresponds to the private key retrieved and at step 8 this unwrapped key is returned to the local computer file system driver. At step 9, the file system driver uses the unwrapped key to decrypt the file after it has been read across the network. Clear data can then be returned to the application on the local computer.

Figure 9 shows how to write an existing encrypted file across a network. Steps 1 to 8 are identical to the read process of figure 14 above. At step 9, the local computer file system driver uses the unwrapped key to encrypt the data being sent to the file before it is transmitted across the network and at step 10 the local application writes clear data to the file. The local computer encrypts that data and sends encrypted data over the network.

CLAIMS

1. A method for encrypting computer files comprising:

encrypting at least one file within a folder;

for each encrypted file, creating an associated key file containing the decryption key,

wherein the association between an encrypted file and its key file is such that any file handling operating performed on the encrypted file is also performed on the key file.

2. A method according to claim 1, comprising marking folders to indicate that files within that folder are to be encrypted.

3. A method according to claim 2, wherein marking a folder comprising placing a file in the folder containing default encryption setting for files created in the folder.

4. A method according to claim 1, 2 or 3, wherein the associated key files are hidden from a user by a file system driver.

5. A method according to claim 4, wherein when an operation is performed on an encrypted file, the file system driver performs the same operations on the key file.

6. A method according to any preceding claim, wherein the key file contains decryption keys for a plurality of users.

7. A method according to any preceding claim, wherein the key file has the same file name as the encrypted file and a different extension.

8. A method according to any preceding claim, wherein the key file comprises a wrapped symmetric key for the associated encrypted file.

9. A method according to claim 8, wherein the symmetric key is wrapped by an end-user=s public key.

10. A method according to claim 3, wherein the default encryption settings includes a list of end-users for which a file is to be encrypted.

11. A method according to claim 10, comprising, on receipt of an encryption create request, the file system driver reads the directory encryption settings file and retrieves the end-user list for which the file needs to be encrypted.

12. A method according to claim 11, wherein the certificates for all the end-users on the list are located.

13. A method according to claims 9 and 12 wherein the key file is created from the symmetric file encryption key is wrapped by the end-user's public key for each end-user in the end-user list.

14. A method according to claim 1, comprising, on receipt of a request to encrypt a file, requesting from the end-user a list of end-users for whom the file is to be encrypted, retrieving the certificates for each end-user on the list, generating a symmetric file encryption key, and wrapping the file encryption key with the certificate for each user, the wrapped encryption keys comprising the key file.

15. A method according to claim 14, wherein encrypting the file comprises saving the file in a temporary file to the name of the key file.

16. A method according to claim 15, comprising requesting the symmetric file key, reading the original clear file, writing the encrypted temporary file, renaming the clear file to a further temporary file, renaming the encrypted temporary file to the original file name and deleting the further temporary file.

17. A method according to claim 3, wherein the directory encryption settings file is stored in a network folder and the file is a network file and wherein encrypted network file is created at a local user computer and sent over the network in encrypted form.

18. A method according to claim 15, wherein the encrypted temporary file is stored in a network folder, and the further temporary file is stored at a local computer, whereby the clear file is not transmitted over the network.

19. A method of decrypting a file encrypted according to the method of any of claims 1 to 18, comprising opening the key file and retrieving the decryption key, retrieving a certificate corresponding to the decryption key, and decrypting the file.

20. A method according to claim 19, wherein the key file comprises a list of users and wrapped keys, and wherein the step of retrieving the list of users from the key file, together with the wrapped keys, retrieving a private key corresponding to the user list, unwrapping the wrapped keys using the private key and using the unwrapped keys to decrypt the file.

21. A method of reading or writing a file encrypted according to the method of any of claims 1 to 20, comprising detecting the key file associated with the encrypted file, retrieving the decryption key from the key file, decrypting the file using the decryption key, and reading and/or writing the decrypted file.

22. A method according to claim 21, wherein retrieving the decryption key comprises finding a private key, and using the private key to unwrap a wrapped key in the key file.

23. A method according to claim 22, wherein the key file includes a list of users for whom the file is encrypted, and finding a private key comprises finding a private key that corresponds to the list of users.

24. A method according to any of claims 21, 22 or 23, wherein the key file is a network key file, the key file is opened at the local user=s computer and the

file is decrypted at the local computer after is has been read across the network.

25. A computer program for encryption and/or decryption of computer files comprising computer code, which when run on a computer or a computer network, performs the steps of any of claims 1 to 24.

26. A computer program for encryption and/or decryption of computer files comprising a program storage medium having recorded thereon computer code, which when run on a computer or a computer network, performs the steps of any of claims 1 to 24.

27. A system for encrypting computer files comprising an installable file system driver transparent to the computer operating system for reading directory encryption settings and retrieving an end user list for which the file is to be encrypted, a daemon, a vault, the daemon operating to retrieve certificates stored in the vault, means for generating a symmetric encryption key, the key being sent by the daemon to the file system driver, the file system driver including means for creating a key file from the end-user certificates and the symmetrical encryption key and for associating the key file with the encrypted file.

28. A system according to claim 27, comprising an object locator, wherein the object locator communicates between the vault and the daemon for retrieval of certificates.

29. A system according to claim 27 or 28, wherein the vault comprises a smart media.

30. A system according to claim 27, 28 or 29 , wherein the vault comprises a soft vault on an end-user=s computer.

31. A system according to any of claims 27 to 30, comprising an operating system shell extension for retrieving certificates, generating the symmetric file encryption key and wrapping the symmetric key with the certificate of each user who has access to an encrypted file.

Application No: GB 0214414.5 **Examiner:** Robin Stout
Claims searched: 1 to 26 **Date of search:** 16 January 2003

Patents Act 1977 : Amended Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance	
X, E	1 at least	WO 02/077747 A2	(XDEGREES) pages 14 to 18.
X	1 at least	EP 1 130 494 A2	(IOMEGA) column 3 line 47 to column 4 line 18.
X	1 at least	EP 1 085 420 A1	(SONY) paragraphs 0247 to 0249.

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC[V]:

G4A

Worldwide search of patent documents classified in the following areas of the IPC[7]:

G06F

The following online and other databases have been used in the preparation of this search report:

EPODOC, WPI, JAPIO